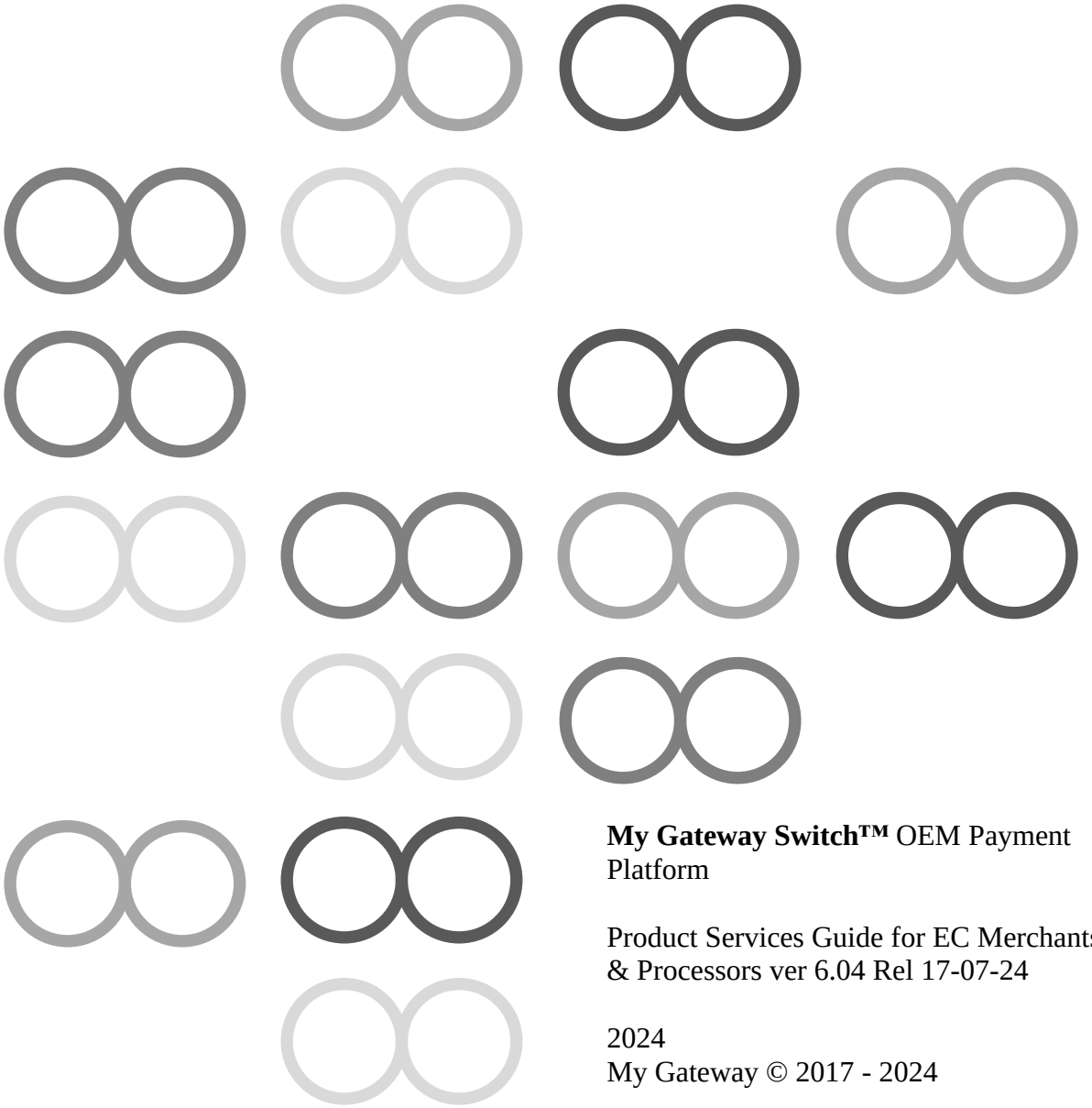
My Gateway

My Gateway™ Product Services Guide for EC Merchants and Processors



My Gateway Switch™ OEM Payment Platform

Product Services Guide for EC Merchants & Processors ver 6.04 Rel 17-07-24

2024

IMPORTANT NOTICES

My Gateway owns the intellectual property in this document exclusively. You acknowledge that you must not perform any act which infringes the copyright or any other intellectual property rights of My Gateway and cannot make any copies of this Manual unless in accordance with these terms and conditions.

Without our express written consent, you must not:

- Distribute any information contained in this Manual to the public media or quote or use such information in the public media; or

- Allow access to the information in this Manual to any company, firm, partnership, association, individual, group of individuals or other legal entity other than your officers, directors and employees who require the information for purposes directly related to your business.

The software described in this Manual is supplied under a license agreement and may only be used in accordance with the terms of that agreement.

My Gateway, My Gateway Switch and the MYGW and My Gateway logos are trademarks of My Gateway.

All third-party product and service names are trademarks or registered trademarks of their respective owners.

SUMMARY OF CHANGES

Not Applicable

1. OVERVIEW 8
My Gateway Switch Benefits 9
My Gateway Switch Description 10
Services Provided to the Merchants 10
2. My Gateway Online Portal 11
3. THE MYGW API INTERFACE 12
The MYGW Security Keys 13
MYGW API Responses 14
Format of a Successful API Request 14
Format of an Unsuccessful API Request 14
4. CLIENT ACCOUNT MANAGEMENT 15
MYGW-CM Interface Calls 16
The Client Data 17
Client Body 19
billingAddress 20
cards 21
The Create Client API Call 22
Client Body - Create Client Request 23
Client Body - Create Client Response 24
Create Client - Sample Source Code 26
The Get Client API Call 27
Client Body- Get Client Request 28
Client Body - Get Client Response 29
Get Client - Sample Source Code 31
The Update Client API Call 32
Client Body - Update Client Request 33
Client Body - Update Client Response 34
Update Client - Sample Source Code 36
The Delete Client API Call 37
Client Body - Delete Client Request 38
Client Body - Delete Client Response 39
Delete Client - Sample Source Code 41

The List Clients API Call 42
Client Body - List Clients Request 43
Client Body - List Clients Response 44
List Clients - Sample Source Code 46
5. PAYMENT MANAGEMENT AND PROCESSING 47
Settlement and Currency Options 48
Settlement 48
Currency 48
MYGW-PM Interface Calls 49
Payment Transaction Flow 49
Email Pay 50
Alipay and WeChat Pay E-Wallets 51
Client Details 51
The Payment Data 52
Payment Body 53
client 55
billingAddress 56
Signature Creation - Sample Source Code 57
The Payment API Call 58
Payment Transaction Request Body 59
Payment Transaction Response 61
Callback URL Processing 63
Payment Request - Sample Source Code 64
The Query Payment API Call 65
Body- Query Payment Request 66
Body – Query Payment Response 67
Query Payment - Sample Source Code 70
Refunds and Cancels /Voids 71
Refunds 71
Cancel/Void 71
Manually Performed Cancel and Refund Transactions 71
The Refund and Cancel API Calls 72

Body – Refund/Cancel Request 73
Refund/Cancel Response 74
Refund/Cancel - Sample Source Code 76

RELATED DOCUMENTATION

The following documents and manuals provide information related to the subjects described in this guide.

- *The Merchant Agreement for Processing Payments (Merchant Agreement)*
- *My Gateway Merchant Administration User Interface Guide (Merchant Admin Guide)*
- *My Gateway Onboarding Guide for EC Merchants and Processors (Setup Guide)*
- *My Gateway Guide to Using UnionPay Cards (Client information)*
- *My Gateway Merchant Enablement Information Form (Merchant Information)*

1. OVERVIEW

My Gateway has implemented a Payment Gateway facility to enable merchants, payment processors and payment service providers to accept common payment cards for Electronic Commerce (EC) and Point of Sale (POS). This document is specific to EC merchants only.

The service is known as the **My Gateway Switch** (or MYGW) and provides internet front-end payment and account management functions and an on-line administrative platform to a web hosted application.

The purpose of the MYGW gateway is to provide a high function payment service that simplifies the payment acceptance process for merchants.

MY GATEWAY SWITCH BENEFITS

The My Gateway Switch provides an outsourced proposition to acquirers, merchants and payment processors (collective referred to as *Payment Service Providers* or PSPs in this document) for this type of service providing:

Significant cost savings – costs incurred for redundant hardware, software, telecommunications and internal operational overheads for an in-house system are avoided.

Faster time to market – MYGW makes it possible for a user to be "live" with their service within a much shorter time frame than usual.

Reduced risk – MYGW clients avoid many of the risks associated with an in-house system such as high initial and on-going costs; recruitment of skilled technical resources; data security requirements and ability to say up to date with the latest industry requirements, compliance and trends.

Business transparency – MYGW is an "out-sourced" packaged service running on Amazon Web Services. All costs are clear to the merchant or processor and agreed up front.

Reduced burden of merchant administration - MYGW provides a web-based merchant administration portal for merchants and processors to manage their payment transactions. The system allows merchants to report on transaction types and perform financial transactions on-line such as refunds, voids and reconciliation.

Wide applicability - MYGW can be used to enable individual merchants regardless of size and number of transactions. It is an ideal solution for the small to medium enterprise (SME) but can also be readily tailored to suit larger enterprises that operate call centres and web sites to service their clients.

Rapid merchant payment enablement - MYGW deploys a flexible connection mechanism based on 'web-services' communication.

Range of payment methods - MYGW currently supports credit and debit card payments from many popular card brands and wallets including UnionPay cards and Alipay mobile applications. Transaction types include Purchases, Void and Refunds.

Secure payments - Data passed between the merchant and MYGW is encrypted using industry standard cryptography. MYGW can reduce the transaction risk by storing card details and other sensitive data on behalf of the merchant.

Future proofing - MYGW will continue to be compliant with new industry initiatives.

MY GATEWAY SWITCH DESCRIPTION

The following diagram describes the My Gateway Switch system and interfaces.

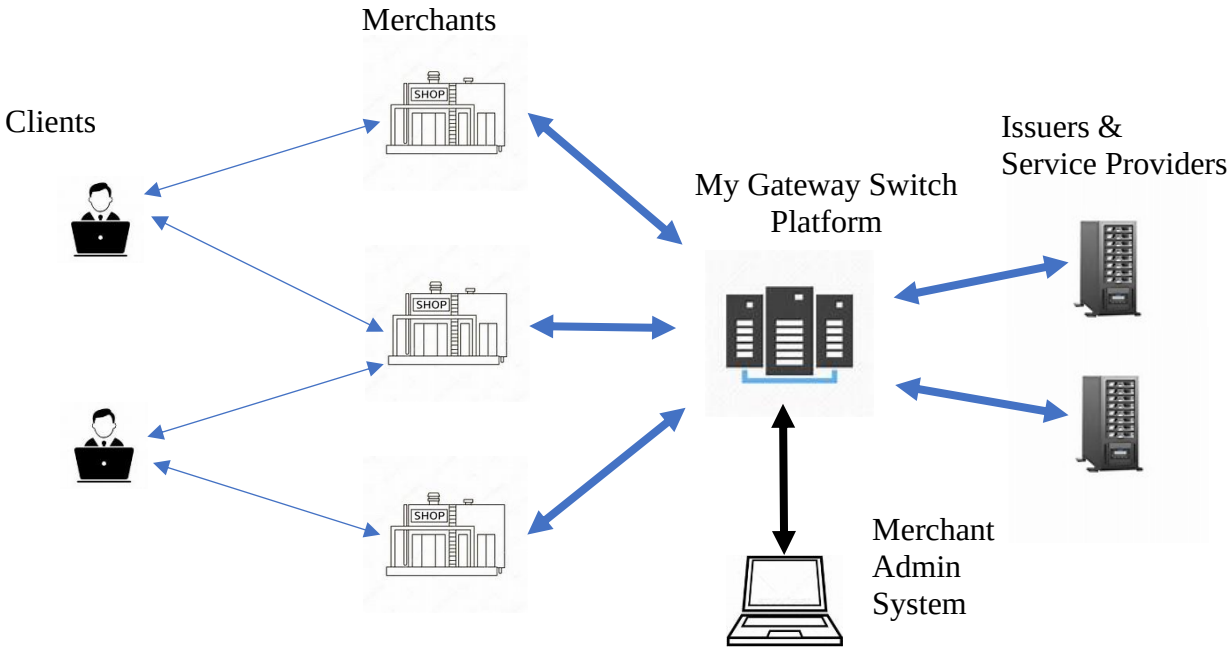


Figure 1. MYGW Architecture

Services Provided to the Merchants

The services provided to merchants and processors by MYGW through the hosted platform are functionally divided into the following groups:

1. Client Account Management
2. Payment Enablement and Processing
3. Merchant Administration Functions

2. MY GATEWAY ONLINE PORTAL

The *Merchant Administration* functions are supplied by providing the merchant or processor with secure browser access to the MYGW central transaction database to perform day-to-day merchant functions. The user is provided with tools for searching, reporting and editing of client accounts and transactions.

The user documentation for the Merchant Administration functions is in a separate document, the *My Gateway Merchant Administration User Interface Guide*.

3. THE MYGW API INTERFACE

The *Client Account Management* and *Payment Management and Processing* services are accessible to the merchant and processors (for EC merchants) using a Web-Services style connection mechanism, where messages are exchanged using JSON encoded resources via RESTful APIs over an encrypted channel. This ensures a simpler integration and allows MYGW to support several integration environments, such as Microsoft COM and .NET, JSP and PHP, etc.

The following sections describe the API requests PSPs and merchants can initiate to manage their clients on the MYGW *Client Account Management* platform, and enable payments through the MYGW *Payment Management and Processing* service.

MYGW maintains two separate systems – test and production, with different URLs. The endpoint for your API request should be set accordingly. While this document will use the web address *oneroadpayments.com*, the actual endpoint address may be different and will be provided by your PSP.

API requests are made by

- sending data to the My Gateway Switch for action, for example to create new clients on the merchant client management system or to initiate payments. These API calls are made using POST or PUT methods.

- requesting data from the My Gateway Switch, for example to retrieve client details from the merchant client management system or query outstanding payment transactions. These API calls are made using the GET method.

THE MYGW SECURITY KEYS

The My Gateway Switch maintains unique security keys for each PSP connected to the system.

The Merchant Key – the merchant key is used solely in payment related requests and is provided to the PSP at sign up. The merchant keys used on the test and production platforms are different and unique and cannot be interchanged. The use of the merchant key and how it is used to validate data transferred between the PSP and the My Gateway Switch is described below.

The Merchant API Key – the merchant API key is used to authenticate the PSP on all API calls made to the My Gateway Switch server, and is provided to the PSP at sign up. This key is common across the test and production platforms for each PSP; but unique to the PSP within the system.

The merchant API key authenticates the origin of each API call through its inclusion in the request header "x-api-key" of the API request.

For example, in JavaScript the usage of the merchant API key may be as follows:

```
// Allocate a Request Object

xhttp = new XMLHttpRequest();

// Set the URL and request headers

xhttp.open( method, endpoint ...);
xhttp.setRequestHeader( "x-api-key", unique_API_key );
```

Both the Merchant Key and the Merchant API key should be kept secure.

MYGW API RESPONSES

All MYGW API Responses include a status code, signifying whether the call succeeded. Additionally, the response will usually include relevant data or a status message and potentially more detailed information.

The data retrieved from the server is in a JSON object.

Format of a Successful API Request

An example of a successful MYGW API request

```
{
"response":
    {
    "statusCode": 201,
    "message": "Successfully Created"
    },
“Optional Data”: {}
}
```

Format of an Unsuccessful API Request

An example of the JSON object sent in response to an unsuccessful MYGW API request is as follows.

```
{
"response":
    {
    "statusCode": 400,
    "message": "Error Message",
    "detail": "Optional More Data"
    }
}
```

An example of executable JavaScript sample source code to manage the data sent from the server in response to an unsuccessful request can be found by clicking on the source code icon below. Note that this is an example of how an error condition can be processed, it is a not a recommendation for how to implement API processing in your own applications.

4. CLIENT ACCOUNT MANAGEMENT

The Client Account Management System or *MYGW Client Management* system (MYGW-CM), is designed for merchants and PSPs who need to manage an on-going relationship with their clients. While merchants and PSPs may provide an account/client management system hosted on their own servers; MYGW-CM provides this service, freeing merchants and PSPs from the need to manage sensitive client data (including potentially payment card information).

MYGW-CM provides each merchant and processor with a database hosted on the MYGW platform. Client information is stored on the merchant's MYGW-CM database and is accessed via Interface Calls from the merchant or processor website.

MYGW-CM INTERFACE CALLS

The following interface calls are available to the merchant or processor from their applications to access the MYGW-CM functions:

1. **Add Client** – used to add a client's details to the merchant's specific MYGW-CM database.
2. **Get Client Information** – used to return a specific client's details stored in the MYGW-CM database including payment transaction history.
3. **Update Client Information** – used to update existing client information or add new client information to an existing client entry in the MYGW-CM database.
4. **Delete Client** – used to remove a client's details from the MYGW-CM database.
5. **List Clients** – used to retrieve a list of all the clients in the MYGW-CM database for a specific merchant.

Note that all these functions can also be manually performed by the merchant and PSP using the merchant administration system described in the My Gateway Merchant Administration User Interface Guide.

Calls are initiated through RESTful APIs to the MYGW-CM system. The URL used to address each of the API calls includes the unique MYGW identifier assigned to the merchant, the merchant ID.

The data required for all client related functions is a *Client Object*, serialised as a JSON string. A MYGW client object holds the details associated with an individual user of the payment system. This user has an association with a given merchant and has been registered as a client of said merchant.

The Client Data

Example of a JSON client Object used to send and receive client data to and from the MYGW server.


```
{
    “name”  : "A Citizen";
    “id” : "6721f4f7-8012-43aa-a7d0-68458a2f9e16";
    “clientID” : "AndrewC";
    “black” : false;
    “correlatedBlack” : false;
    “merchantID” : "8873abde-7b40-4ac2-8774-be331fa80ffd ";
    “merchantName” : "The Merchant";
    “mobileNumber” : "+610411123456";
    “clientPhone” : "+61212345678";
    “emailAddress” : "client@email.com";
    “created” : "2017-10-05T02:23:52.645Z ";
    “updated” : "2017-10-17T21:36:50.575Z ";
    “valid”  : true;
    "billingAddress" : {
        "streetNumber" : "23",
        "streetName" : "High",
        "streetSuffix" : "St",
        "city" : "Sydney",
        "state ": "NSW",
        "postCode" : "2000",
        "country" : "Australia",
        "valid" : true
    },
    "cards" : {
        "pan" : "************0064",
        "expiry" : "MM/YY",
        "name" : "Andrew Citizen",
        "expiryMonth" : "MM",
        "expiryYear ": "YYYY",
        “default” : true,
        “valid” : true;
        },
        {
        "pan" : "************0065",
        "expiry" : "MM/YY",
        "name" : "A Citizen",
```

```
    "expiryMonth" : "MM",
    "expiryYear" : "YYYY",
    “default” : false,
    “valid” : true
    }
};
```

Not all the elements of the object are required for every MYGW-CM API call and some are set by the MYGW server. The following tables describe which elements are required for some calls, are optional or read only (as described below in the specific API calls).

R: System supplied, read-only.
M: Mandatory.
O: Optional.

Client Body

Name	Description	R/M/O
id	Gateway identifier, supplied by the MYGW server	R
Name	Client's name	M
clientID	Key or name to identify the customer	M
black	Flags whether the client is blacklisted. These clients cannot complete transactions until the blacklist is lifted. Set by the MYGW system	R
correlatedBlack	Flags whether a client has some correlation (e.g. same email address, phone or billing address) as a blacklisted client. These clients cannot complete transactions until the blacklist is lifted. Set by the MYGW system	R
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
merchantName	The merchant's name as stored on the MYGW server	O
mobileNumber	The client's mobile phone number which may be used to securely communicate with, and identify the client	M
clientPhone	Alternative phone number of the client	O
emailAddress	Valid email address of the client, used to contact the client if necessary	M
created	Date and time when the client record was created, determined by the MYGW system	R
updated	Date and time when the client record was last updated, determined by the MYGW system	R
valid	Flags whether the client has been validated, set by the MYGW system	R

billingAddress

The client's address may optionally be provided when the client is created; in which case the details will be returned when the client's data is accessed.

Name	Description	R/M/O
streetNumber	The number on a street of the client's postal address	O
streetName	the name of the street of the client's postal address	O
streetSuffix	The tail of the street address, of the client's postal address, typically St, Rd, etc	O
city	The town, city or village that is used in the client's postal address	O
state	The state or region used in the client's postal address	O
postcode	The postal code used in the client's postal address	O
country	The country used in the client's postal address	O
valid	Flags whether the client address has been validated, set by the MYGW system	R

cards

Each client may optionally have up to 3 account details (card details) provided when the client is created, These 'lodged' accounts are referred to as *Cards on File* and may expedite payment processing by reducing the need for account information to be entered for every payment transaction. If set, this information will be returned when the client's data is accessed.

Name	Description	R/M/O
pan	The Primary Account Number, or that embossed on the front of the credit card	O
expiry	The expiration date of the credit card.	O
name	The name of the person that the credit card was issued to.	O
expiryMonth	This is a derived field, and is the month of the card expiration date.	R
expiryYear	This is a derived field, and is the year of the card expiration date.	R
default	This field indicates that this is the default card, if more than one card is lodged	O
valid	Flags whether the account details have been validated, set by the MYGW system	R

An example of executable JavaScript sample source code to create a client object can be found by clicking on the source code icon below. Note that this is an example of how a client object is structured, it is a not a recommendation for how to implement the client APIs in your own applications.

The Create Client API Call

This call is used to add a new client's details to the merchant's specific MYGW-CM database.

URLs:

Test
https://api. test.oneroadpayments.com/merchants/{*merchantID*}/clients
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/clients

Request Method:

POST

Headers:

The header '*x-api-key*' should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	R/M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M

Client Body - Create Client Request

Name	Description	R/M/O
Name	Client's name	O
clientID	Key or name to identify the customer	M
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
merchantName	The merchant's name as stored on the MYGW server	O
mobileNumber	The client's mobile phone number which may be used to securely communicate with, and identify the client	M
clientPhone	Alternative phone number of the client	O
emailAddress	Valid email address of the client, used to contact the client if necessary	M
billingAddress	The client's postal or home address, formatted as shown above	O
cards	Payment or card details of up to 3 accounts, formatted as shown above	O

The *client data* should be sent to the MYGW server formatted as a JSON object, using the POST method.

Client Body - Create Client Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful create client request is as follows.


```
{
"response":
    {
    "statusCode": 201,
    "message": "Successfully Created"
    },
"client":
{
"merchantID": "9678h65fr-fb52-4a8a-8de6-438ecf8d1104",
"merchantName": "The Merchant",
"mobileNumber": "2244668800",
"clientID": "NewClient",
"created": "2024-07-13T08: 46: 39.347",
"updated": "2024-07-13T08: 46: 39.347",
"name": "NewClient",
"id": "a0f58b93-0336-458f-8111-d19aaf1b57e8",
"emailAddress": "newclient@client.org",
"black": false,
"billingAddress":
    {
    "streetNumber": "",
    "streetName": "",
    "streetSuffix": "",
    "city": "",
    "state": "",
    "country": "",
    "valid": false
    },
    "valid": true,
    "correlatedBlack": false
    }
}
```


The response will include a complete client object, as described above – *The JSON Client Object*, even if some of those fields (such as *cards* or *billingAddress*) are not set.

An example of the JSON object sent in a response to an unsuccessful create client request is as follows.


```
{
```

```
"response":
   {
   "statusCode": 400,
   "message": "Bad Request",
   "detail": "Invalid Merchant ID"
   }
}
```

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200/201	The new client was successfully created at the server	*Message* - Successfully Created
400	The merchant ID was invalid or not recognised, and the request failed	*Message* - Bad Request *Detail* – Invalid Merchant ID
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
409	A client with the specified details already exists at this merchant, and the request failed	*Message* – Conflict *Detail* – ClientID already exists for merchant
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Create Client - Sample Source Code

An example of executable JavaScript sample source code to create a client can be found by clicking on the source code icon below. Note that this is an example of how a create client request can be initiated, it is a not a recommendation for how to implement the client APIs in your own applications.

The Get Client API Call

This call is used to return a specific client's details stored in MYGW-CM.

URLs:

Test
https://api.test.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}

Request Method:

GET

Headers:

The header *'x-api-key'* should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	R/M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
clientID	Uniquely identifies a client within a merchant's registered client collection. This can be either the *clientID* or *id* parameters detailed in the *client* object above	M

Client Body- Get Client Request

NA- No client body data required.

The get client request should be sent to the MYGW server using the GET method.

Client Body - Get Client Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful get client request is as follows.


```
{
"response":
    {
    "statusCode": 200,
    "message": "OK"
    },
"client":
{
"merchantID": "9678h65fr-fb52-4a8a-8de6-438ecf8d1104",
"merchantName": "The Merchant",
"mobileNumber": "2244668800",
"clientID": "ACit123456",
"created": "2024-07-13T08: 46: 39.347",
"updated": "2024-07-13T08: 46: 39.347",
"name": "Andrew Citizen",
"id": "a0f58b93-0336-458f-8111-d19aaf1b57e8",
"emailAddress": "acitizen@client.org",
"black": false,
"billingAddress":
    {
    "streetNumber": "88",
    "streetName": " New ",
    "streetSuffix": " Street ",
    "city": "Newtown",
    "state": "NSW",
    "country": "Australia",
    "valid": true
    },
    "valid": true,
    "correlatedBlack": false
    }
}
```


The response will not necessarily include a complete client object (as described above – *The JSON Client Object*), only fields that were included when the client was created or updated on the system. If some of those fields (such as *cards* or *billingAddress*) are not set, they will not be returned from the host.

An example of the JSON object sent in a response to an unsuccessful get client request is as follows.

```
{
"response":
    {
    "statusCode": 404,
    "message": " "Not Found",
    "detail": " No Client found with MerchantID: 9678h65fr-fb52-4a8a-8de6-
438ecf8d1104 and ClientID: A Citizen"
    }
}
```


The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The client details were successfully returned from the server	*Message* - OK
400	The request was not correctly formatted	*Message* - Malformed request *Detail* – (e.g) missing path parameters
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
404	No client with the ID specified was found for the merchant at the server	*Message* – Not Found *Detail* – No Client found with MerchantID and ClientID
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Get Client - Sample Source Code

An example of executable JavaScript sample source code to get a client's data can be found by clicking on the icon below. Note that this is an example of how a get client request can be initiated, it is a not a recommendation for how to implement the client APIs in your own applications.

The Update Client API Call

This call is used to update an existing client's information or add new client information to an existing client entry in the MYGW-CM database.

URLs:

Test
https://api.test.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}

Request Method:

PUT

Headers:

The header '*x-api-key*' should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	R/M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
clientID	Uniquely identifies a client within a merchant's registered client collection. This can be either the *clientID* or *id* parameters detailed in the *client* object above	M

Client Body - Update Client Request

Name	Description	R/M/O
Name	Client's name	O
clientID	Key or name to identify the customer	M
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
merchantName	The merchant's name as stored on the MYGW server	O
mobileNumber	The client's mobile phone number which may be used to securely communicate with, and identify the client	M
clientPhone	Alternative phone number of the client	O
emailAddress	Valid email address of the client, used to contact the client if necessary	M
billingAddress	The client's postal or home address, formatted as shown above	O
cards	Payment or card details of up to 3 accounts, formatted as shown above	O

The new or updated *client data* should be sent to the MYGW server formatted as a JSON object, using the PUT method.

Client Body - Update Client Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful update client request is as follows.


```
{
"response":
    {
    "statusCode": 200,
    "message": "OK"
    },
"client":
{
"merchantID": "9678h65fr-fb52-4a8a-8de6-438ecf8d1104",
"merchantName": "The Merchant",
"mobileNumber": "2244668800",
"clientID": "NewClient",
"created": "2024-07-13T08: 46: 39.347",
"updated": "2024-07-13T08: 46: 39.347",
"name": "NewClient",
"id": "a0f58b93-0336-458f-8111-d19aaf1b57e8",
"emailAddress": "newclient@client.org",
"black": false,
"billingAddress":
    {
    "streetNumber": "Updated Street Number",
    "streetName": " Updated Street",
    "streetSuffix": "Updated Suffix",
    "city": "Updated City",
    "state": "Updated State",
    "country": "Updated Country",
    "valid": false
    },
    "valid": true,
    "correlatedBlack": false
    }
}
```


The response will not necessarily include a complete client object (as described above – *The JSON Client Object*), only fields that were included when the client was created or updated on the system. If some of those fields (such as *cards* or *billingAddress*) are not set, they will not be returned from the host.

An example of the JSON object sent in a response to an unsuccessful create client request is as follows.

```
{
"response":
   {
   "statusCode": 400,
   "message": "Bad Request",
   "detail": "Invalid Merchant ID"
   }
}
```


The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The client details were successfully updated at the server	*Message* - OK
400	The request was not correctly formatted	*Message* - Malformed request *Detail* – (e.g) missing path parameters
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
404	No client with the ID specified was found for the merchant at the server	*Message* – Not Found *Detail* – No Client found with MerchantID and ClientID
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Update Client - Sample Source Code

An example of executable JavaScript sample source code to update a client's data can be found by clicking on the icon below. Note that this is an example of how an update client request can be initiated, it is a not a recommendation for how to implement the client APIs in your own applications.

The Delete Client API Call

This call is used to remove a client's details from the MYGW-CM database.

URLs:

Test
https://api.test.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/clients/{clientID}

Request Method:

DELETE

Headers:

The header *'x-api-key'* should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	R/M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
clientID	Uniquely identifies a client within a merchant's registered client collection. This can be either the *clientID* or *id* parameters detailed in the *client* object above	M

Client Body - Delete Client Request

NA – No client body data required.

The *delete request* should be sent to the MYGW server using the DELETE method.

Client Body - Delete Client Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful update client request is as follows.


```
{
"response":
    {
    "statusCode": 200,
    "message": "OK"
    },
"client":
    {
    "merchantID": "9678h65fr-fb52-4a8a-8de6-438ecf8d1104",
    "merchantName": "The Merchant",
    "mobileNumber": "2244668800",
    "clientID": "TheClient",
    "created": "2024-07-13T08: 46: 39.347",
    "updated": "2024-07-13T08: 46: 39.347",
    "name": "The Client",
    "id": "a0f58b93-0336-458f-8111-d19aaf1b57e8",
    "emailAddress": "theclient@client.org",
    "black": false,
    "valid": true,
    "correlatedBlack": false
    }
}
```


The response will not necessarily include a complete client object (as described above – *The JSON Client Object*), only fields that were included when the client was created or updated on the system. If some of those fields (such as *cards* or *billingAddress*) are not set, they will not be returned from the host.

An example of the JSON object sent in a response to an unsuccessful create client request is as follows.


```
{
"response":
    {
    "statusCode": 400,
    "message": "Bad Request",
    "detail": "Invalid Merchant ID"
    }
}
```


The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The client details were successfully deleted from the server	*Message* - OK
400	The request was not correctly formatted	*Message* - Malformed request *Detail* – (e.g) missing path parameters
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
404	No client with the ID specified was found for the merchant at the server	*Message* – Not Found *Detail* – No Client found with MerchantID and ClientID
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Delete Client - Sample Source Code

An example of executable JavaScript sample source code to delete a client can be found by clicking on the icon below. Note that this is an example of how a delete client request can be initiated, it is a not a recommendation for how to implement the client APIs in your own applications.

The List Clients API Call

This call is used to retrieve a list of all the clients in the MYGW-CM database for a specific merchant.

URLs:

Test
https://api.test.oneroadpayments.com/merchants/{*merchantID*}/clients?next=*{next client}*
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/clients?next=*{next client}*

Request Method:

GET

Headers:

The header '*x-api-key*' should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	R/M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
next	The client ID of the client to start downloading the client list from	O

Client Body - List Clients Request

NA – No client body data required.

The *list clients request* should be sent to the MYGW server using the GET method.

Client Body - List Clients Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful list clients request is as follows.


```
{
"response":
   {
   "statusCode":200,"message":"OK"
   },
"clients":[
   {
   "merchantID":"1234abcd-5678-efgh-9012-123456abcdef",
   "merchantName":"The Merchant",
   "mobileNumber":"222333444555",
   "clientID":"TheClient_1",
   "created":"2023-11-06T06:46:53.953",
   "updated":"2024-07-14T08:18:22.747",
   "name":"First Client",
   "id":"16639678-2c4f-481b-bda6-3144cf9ae8fb",
   "emailAddress":"client_1@client1.com",
   "black":false,
   "valid":true,
   "correlatedBlack":false
   },
   {
   "merchantID":"1234abcd-5678-efgh-9012-123456abcdef",
   "merchantName":"The Merchant",
   "mobileNumber":"666777888",
   "clientID":"TheClient_2",
   "created":"2023-11-06T08:06:26.065",
   "updated":"2024-07-14T06:51:57.540",
   "name":"Second Client",
   "id":"826f55b0-4f7d-4700-b53c-b616bceef3f6",
   "emailAddress":"client_2@client2.com",
   "black":false,
   "valid":true,
   "correlatedBlack":false
   },
   {
  // A record in the clients array for each client stored for this merchant
     .....
   }
   ],
“next”: “nextClientID”
}
```

A successful response will include in the response JSON object a list of every client stored at the server for the merchant. Each client object will include all the information stored at the server for that client, in the format described above – *The JSON Client Object*.

However, the individual client information will not necessarily include a complete client object, only fields that were included when the client was created or updated on the system. If some of those fields (such as *cards* or *billingAddress*) are not set, they will not be returned from the host.

If the *next* element is included in the response, it signifies that there are clients which have still not been loaded, most probably because the client list is too long to send in a single response. The *next* element contains the client ID of the first client in the remainder of the list. To retrieve the rest of the client list, the *list clients* API request should be called again, with the URL parameter 'next' set to the next client ID. To retrieve the entire list, this should be done repeatedly until the *next* parameter in the returned JSON object is not included.

An example of the JSON object sent in a response to an unsuccessful create client request is as follows.

```
{
"response":
   {
   "statusCode": 400,
   "message": "Bad Request",
   "detail": "Invalid Merchant ID"
   }
}
```

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The client list was successfully loaded from the server	*Message* - OK
400	The merchant ID was invalid or not recognised, and the request failed	*Message* - Bad Request *Detail* – Invalid Merchant ID
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
500	A server error occurred, and the request failed	*Message* – Internal processing exception

List Clients - Sample Source Code

An example of executable JavaScript sample source code to list a merchant's clients can be found by clicking on the icon below. Note that this is an example of how a list clients request can be initiated, it is a not a recommendation for how to implement the client APIs in your own applications.

5. PAYMENT MANAGEMENT AND PROCESSING

The Payment Management and Processing or MYGW Payment Management system (MYGW-PM) supports all Chinese credit and debit card brands; and the most popular Chinese e-wallets.

MYGW-PM uses a Server Hosted architecture for EC merchants where the client has the option to input their account or card details directly at the merchant or on a 'redirect' page hosted by the My Gateway Switch. Some payment methods only support the redirect method, including UnionPay and the Chinese e-wallets, in which case the merchant or PSP does not need to take responsibility for securing card details and only needs to provide MYGW-PM with the payment amount and client ID for any transaction.

MYGW-PM transactions use the SSL protocol to provide secure transmission of sensitive data between a client's web browser and the MYGW-PM system.

SETTLEMENT AND CURRENCY OPTIONS

Settlement

Settlement is the process whereby funds are transferred to merchants' bank accounts from their PSP. Merchant funds are transferred into their bank account after their transactions have been authorised by the card issuing institution and funds have been cleared by the relevant card scheme network. Settlement usually takes place a number of days after the transactions have been approved, the 'Settlement Date'. The settlement date is generally agreed between the merchant and their PSP prior to enablement.

Currency

MYGW supports multiple currencies for transactions and supports PSPs to settle in the currency of their choice. The My Gateway Switch allows merchants and PSPs to specify options for :

1. **Presentment Currency** – This is the currency the merchant or PSP uses to send transactions to the My Gateway Switch, and is specified in the payment API request as described below. Merchants and PSPs may choose to present in more than one currency – the My Gateway Switch is a multi-currency platform.
2. **Settlement Currency** – This is the currency used to settle transactions to the PSPs and merchants. It is the currency of the funds deposited into merchant bank accounts as described above.
3. **Billing Currency** – This is the currency the PSP will bill the merchant in and the currency the PSP fees are denominated in.

MYGW-PM INTERFACE CALLS

The following interface calls are available to the merchant or processor from their platform to access the MYGW-PM functions:

1. **Purchase** – used to initiate a payment process to allow a client to pay for goods and services provided by the merchant using their debit and credit cards or e-wallets.
2. **Query** – used to establish the status of a previously executed Purchase.
3. **Void/Purchase Cancellation** – used to cancel a purchase made prior to the merchant being settled for the purchase.
4. **Refund** – used to refund money to a client's card after the merchant has been settled for the purchase.

Note that the Void and Refund transactions can also be manually performed by the merchant using the merchant administration system described the My Gateway Merchant Administration User Interface Guide.

While the Purchase transaction results in the client being directed to a web page served from the merchant, processor or MYGW-PM platform; the Void and Refund transactions can be processed by the system without client input.

A detailed explanation of how UnionPay cards, and Alipay and WeChat Pay wallets are used and processed on the MYGW-PM system is provided in the document *Using Chinese Payments*. The following contains a more high-level overview of that process and should be read in conjunction that document.

Whether a client has been registered as a client on the MYGW-PM system or not by the merchant, the typical steps in a MYGW-PM payment transaction are as follows:

Payment Transaction Flow

1. The client selects a commodity or service on the merchant's website. The merchant then initiates a payment transaction request to the MYGW-PM system using an API request, potentially including the existing or a new client ID and payment amount.

2. MYGW-PM confirms the merchant and client identification and starts the payment process.

3. MYGW-PM navigates the cardholder's browser to the payment page. On this page, the client may be provided with a QR Code and the client may scan this code using the UnionPay, Alipay or WeChat Applications. Alternatively, the client enters payment information including card number, card verification code and expiry date. In addition, clients using some payment cards may input their cell phone number and perform an additional verification process.

Figure 2. Example QR Code

4. MYGW-PM then packages the payment information and sends a payment request message to the card issuer.

5. The issuer processes the transaction and responds to MYGW-PM after completing authorization.

6. MYGW-PM receives the transaction result from the issuer and informs the client of the result of the transaction, via a callback to the client browser.

7. MYGW-PM then sends the transaction result to the merchant. The merchant can then update its order information and if necessary, provide the relevant goods or services to the client.

Email Pay

Email Pay is a more secure payment method implemented for some merchants connected to the My Gateway system. This payment method is currently only available for UnionPay cards and wallets; and requires the client to have a valid registered email address stored in their client profile at the MYGW client management system.

1. The client selects a commodity or service on the merchant's website. The merchant then initiates a request to the MYGW-PM system using an API request and including the new or existing client ID and payment amount.

2. MYGW-PM confirms the merchant and client identification and starts the enablement process.

3. MYGW-PM sends a payment enablement email to the client's registered email address, or the email address provided for the newly created client.

4. The response to this call will indicate whether the email has been sent to the client's email address successfully – a response of *210* means the email has been sent.

5. Once the client receives the email and follows the instructions contained therein, the payment process is similar to the one described above under *Payment Transaction Flow*, from step 3.

Alipay and WeChat Pay E-Wallets

1. The client selects a commodity or service on the merchant's website. The merchant then initiates a request to the MYGW-PM system using an API request and including the existing or new client ID and payment amount.

2. MYGW-PM confirms the merchant and client identification and starts the payment process.

3. MYGW-PM navigates the cardholder's browser to the payment page. On this page, the client is presented with a unique QR Code and information on the payment amount and payment currency.

4. The client then scans the QR Code using their wallet application payment function, the amount of the transaction is then presented to the client who is asked to approve the transaction and enter their payment password on the Application.
The payment information is sent in a payment request message to the card issuer.

5. The issuer processes the transaction and if approved a 'Success' message is displayed on the client Application with the amount and a response is sent to MYGW-PM after completing authorization.

6. MYGW-PM receives the transaction result from the issuer and informs the client of the result of the transaction.

7. MYGW-PM then sends the transaction result to the merchant. The merchant can then update its order information and provide the relevant goods or services to the client.

The payment interface calls are initiated in the same way as the previously described client management APIs. Described below are the details of the payment management API calls.

Client Details

The client, the person making the purchase at the merchant's website, could be a guest (i.e. a one time or sporadic customer), a client managed by the merchant themselves or a client with a record stored on the MYGW-CM server, as described above.

Some PSPs and merchants will be required to only initiate payment transactions for clients who have been added to the MYGW-CM server. In this case the payment request will either need to include a valid client ID of an existing client or the details of a new client. In the latter case a new client with the information provided will be created at the server, prior to the payment transaction being initiated.

The Payment Data

Example of a JSON Payment Object used to send and receive payment data to and from the MYGW server.


```
{
"clientID":"NewClient",
"amount":"100",
"cardType":"UnionPay",
"currency":"USD",
"echo":"",
"orderID":"088345853",
"replyURL":"https://www.my-gateway.net/ReturnURL.html",
"backofficeURL":"https://www.my-gateway.net/returl.php",
"sendEmail":0
"signature":"ffeee47ce763fa8ca9129d6ded4c9da0",
"signatureVersion":"1.0",
"client":
   {
   "clientID":"NewClient",
   "mobileNumber":"+610411123456",
   " clientPhone ":"+6121234567890",
   "name":"A Citizen",
   "emailAddress":" client@email.com ",
   "billingAddress":
      {
      "streetNumber":"100",
      "streetName":"New",
      "streetSuffix" : "St",
     "city":"Newtown",
      "state":"NSW",
      "postCode":"8888",
      "country":"Australia"
      }
   },
}
```


Not all the elements of the object are required for every MYGW-PM API call and some are supplied by the MYGW server. The following tables describe which elements are mandatory for some calls and which are optional (as described below in the specific API calls).

M: Mandatory.
O: Optional.

Payment Body

Name	Description	M/O
clientID	Key or name to identify the customer, depending on the merchant this may be managed by the merchant or by the MYGW-CM system. Some merchants are required to include a valid clientID or new client object	O
amount	The amount of the purchase transaction	M
cardType	The type of payment card or account to be used for the transaction. Depending on the merchant this can be one of 'UnionPay', 'Alipay' or 'Wechat Pay'	M
currency	The transaction currency. This must match a currency setup for the merchant at sign up	M
echo	Value echoed back at the point of notification. This is not used during transaction authorization	O
orderID	The unique merchant order number for this transaction. This is used by the merchant or PSP to identify transactions with the MYGW server if required	M
replyURL	The 'front end' URL, called by MYGW-PM once the transaction completes with the result of the transaction. This URL will be displayed to the client to inform them of the result of the transaction. If this field is not set or left blank, the URL stored on the server for the merchant/PSP will be used	O
backofficeURL	The merchant or PSP back-end server URL called when a transaction completes with the result of the transaction. If this field is not set or left blank, the URL stored on the server for the merchant/PSP will be used	O
sendEmail	Flags whether the *Email Pay* process (see above) should be used to process this transaction. This is only relevant is the merchant is registered for Email Pay for this cardType	O

signature	The cryptographic hash for this payment transaction, created using the *Merchant Key*, used to authenticate the payment request	M
signatureVersion	Indicates the version of the signature to authenticate this transaction. Currently ‘1.0’	M
client	If a new client is being created for this transaction, contains the client object described below	O

client

The client data to be included in the *client* field of the payment request, should a merchant require that a new client be created at the MYGW-CM server.

Name	Description	M/O
Name	Client's name	O
clientID	Key or name to identify the customer	O
mobileNumber	The client's mobile phone number which may be used to securely communicate with, and identify the client	O
clientPhone	Alternative phone number of the client	O
emailAddress	Valid email address of the client, used to contact the client if necessary	O
created	Date and time when the client record was created, determined by the MYGW system	O
updated	Date and time when the client record was last updated, determined by the MYGW system	O
billingAddress	The client's street or postal address, described below	O

billingAddress

The client's address may optionally be provided when the client is created; in which case the details will be stored with the client record.

Name	Description	R/M/O
streetNumber	The number on a street of the client's postal address	O
streetName	the name of the street of the client's postal address	O
streetSuffix	The tail of the street address, of the client's postal address, typically St, Rd, etc	O
city	The town, city or village that is used in the client's postal address	O
state	The state or region used in the client's postal address	O
postcode	The postal code used in the client's postal address	O
country	The country used in the client's postal address	O

An example of executable JavaScript sample source code to create a payment object can be found by clicking on the source code icon below. Note that this is an example of how a payment object is structured, it is a not a recommendation for how to implement the client APIs in your own applications.

Signature Creation - Sample Source Code

The signature field of the payment request is required to validate that the payment request originated from a valid merchant or PSP. The signature is created using unique transaction data and includes the Merchant Key allocated at sign up.

The signature uses the MD5 message-digest hash function and the following transaction data:

- The *clientID*
- The *orderId*
- The *amount*
- The *currency*
- The *replyURL*
- The *backofficeURL*

The signature is created by concatenating the lower-case data above; including the MD5 hash of the Merchant Key; and storing the signature as the MD5 hash of the resulting string.

An example of executable JavaScript sample source code to create the signature can be found by clicking on the icon below. Note that this is an example of how a signature can be created, it is a not a recommendation for how to implement the payment API in your own applications.

The Payment API Call

This call is used to initiate a payment from a merchant or PSP. Once the payment is initiated, in most cases further client or user input is required. The payment will be completed by the user, and the result returned to the merchant or PSP via the callback URLs.

If callback URLs are included in the payment request, they will be used. If callback URLs are not included in the payment request, the default URLs for the merchant or PSP, stored at the My Gateway Switch when the merchant was added will be used.

URLs:

Test
https://api.test.oneroadpayments.com/merchants/{*merchantID*}/transactions
Production
https://api.oneroadpayments.com/merchants/{*merchantID*}/transactions

Request Method:

POST

URL Parameters:

Name	Description	M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M

Payment Transaction Request Body

Name	Description	M/O
clientID	Key or name to identify the customer, depending on the merchant this may be managed by the merchant or by the MYGW-CM system. Some merchants are required to include a valid clientID or new client object	O
amount	The amount of the purchase transaction	M
cardType	The type of payment card or account to be used for the transaction. Depending on the merchant this can be one of 'UnionPay', 'Alipay' or 'Wechat Pay'	M
currency	The transaction currency. This must match a currency setup for the merchant at sign up	M
echo	Value echoed back at the point of notification. This is not used during transaction authorization	O
orderID	The unique merchant order number for this transaction. This is used by the merchant or PSP to identify transactions with the MYGW server if required	M
replyURL	The 'front end' URL, called by MYGW-PM once the transaction completes with the result of the transaction. This URL will be displayed to the client to inform them of the result of the transaction. If this field is not set or left blank, the URL stored on the server for the merchant/PSP will be used	O
backofficeURL	The merchant or PSP back-end server URL called when a transaction completes with the result of the transaction. If this field is not set or left blank, the URL stored on the server for the merchant/PSP will be used	O
sendEmail	Flags whether the *Email Pay* process (see above) should be used to process this transaction. This is only relevant is the merchant is registered for Email Pay for this cardType	O

signature	The cryptographic hash for this payment transaction, created using the *Merchant Key*, used to authenticate the payment request	M
signatureVersion	Indicates the version of the signature to authenticate this transaction. Currently '1.0'	M
client	If a new client is being created for this transaction, contains the client object described above	O

The *payment request* should be sent to the My Gateway Switch server using the POST method.

In most cases this call is used to initiate a payment and as described above, this will result in the client or user being redirected to a separate transaction page to complete payment. This can be done by entering card or account details; scanning a QR code and/or entering an authentication code (in the case of 3DS for example) to complete the payment.

Merchants and PSPs are notified of the result of the payment transaction by way of the callback URLs described in the payment fields above.

Payment Transaction Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

This response relates to the payment API call; a success response does not mean the payment transaction has completed successfully, only that the payment initiation process has successfully started.

The final result of the payment transaction is sent using the callback URLs.

The response to the payment request includes in the field *response* HTML code to redirect the user to the required payment page.

An example of the data sent in a response to a successful payment request is as follows.

{
status: 200,
response:”<?xml version="1.0" encoding="UTF-8"?><html class="mac chrome chrome5 webkit webkit5"><head><base href="/"><meta http-equiv="refresh" content="0;url=https://payments.oneroadpayments.com/redirect2.html?curr=036&merc=SE&id=tc8d94856-db35-4f85-b7c7-e96a0b11d5d5&stage=test&amt=10000&m=5adeaafb-1b6d-4bb2-ba11-1cce35e6b38e&o=108029257&sign=027ae8d96fc5e41d3e79372efd1f4e24&time=20240716133512&order=20240716969339133512"/></head><body></body></html>”
}

Rendering the code in the *response* field will redirect the client or user to the payment page to complete the payment transaction.

An example of the JSON object sent in a response to an unsuccessful payment request is as follows.


```
{
"response":
   {
   "statusCode":400,
   "message":"Bad Request",
   "detail":"Amount out of range - for this merchant; USD [min = 1.00, max = 15000.00]"
   }
}
```


An example of how to process error conditions is included above under *WASP API Responses*.

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The payment request was successfully initiated, and the response includes the redirect HTML code	*Message* - NA
210	The payment request was successfully initiated, and a payment enablement email has been sent to the client. Used with *Email Pay*	*Message* - NA
400	The transaction amount is out of range for the merchant. Transaction ranges are allocated to merchants at sign up	*Message* - Bad Request *Detail* – Amount out of range for this merchant
400	The presenting currency is not supported for this merchant. Merchants are allocated a set of currencies at sign up	*Message* - Bad Request *Detail* – Currency not supported for this merchant
400	The signature included in the request could not be verified using the Merchant Key	*Message* - Bad Request *Detail* – Signature mismatch
404	The merchant ID was invalid or not recognised	*Message* – Not Found *Detail* –Merchant not found
404	A client ID was included in the payment request, and it could not be recognised as belonging to the merchant	*Message* – Not Found *Detail* –Client record not found
422	The order number in the payment request has been used with a previous transaction and the order cannot be completed	*Message* – Unprocessable Entity *Detail* –Duplicate Order ID
422	No path to the requested payment system could be found for this transaction or merchant. This error most	*Message* – Unprocessable Entity

	often occurs when a merchant has not been enabled for the *cardType* specified in the payment request.	*Detail* – Unable to determine gateway processor
422	The merchant has exceeded their daily, weekly, fortnightly or monthly transaction or volume limit. Merchants are allocated their limits at sign on	*Message* – Unprocessable Entity *Detail* – Blocked due to risk score *Reason* - Exceed transaction attempts
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Note: Other reasons for the payment request failing may include risk management parameters, limits or ranges being exceeded. In every case the status will be *422* and the response will include *message*, *detail* and optionally *reason* fields.

When any of the errors listed above are returned (i.e. the status codes except *200* and *210*), it means that the transaction has **not** been initiated and there is no further action taken with that transaction. In this case, no callback is sent to the reply or backend URL and processing of the transaction is complete.

Callback URL Processing

When the response to the payment request is either *200* or *210*, it means that the transaction has been initiated and will complete at some time in the future. The final result of this transaction will be notified to the client or user via a callback to the *replyURL* and to the merchant or PSP via a callback to the *backofficeURL*.

The structure of the URL callback is the same for the *replyURL* and the *backofficeURL,* as follows for a successful transaction

https://*{callbackURL}*?currency=AUD&success=Y&merchantID=5adeaafb-1b6d-4bb2-ba11-1cce35e6b38e&orderID=110836419&clientID=NewClient&amount=25.00&signature=3a53e1e7251b08036cc2f9b8de9d2030

and as follows for a failed transaction

https://*{callbackURL}*?success=N&merchantID=5adeaafb-1b6d-4bb2-ba11-1cce35e6b38e&orderID=110836419&clientID=NewClient&detail=Transaction+abandoned&signature=1ef02c01f953aec4163bc06d4c287c6f

The *signature* URL parameter is used to verify that the callback originated from the My Gateway Switch. The signature is generated in a similar way to the signature included in the payment request, except that the merchant ID is appended to the data

to be signed. The same Merchant Key is used to generate and verify the signature in the callback URL.

The callback is sent using the GET method.

An example of executable JavaScript sample source code to verify the URL signature can be found by clicking on the icon below. Note that this is an example of how a URL signature can be verified, it is a not a recommendation for how to implement callbacks in your own applications.

Payment Request - Sample Source Code

An example of executable JavaScript sample source code to initiate a payment can be found by clicking on the icon below. Note that this is an example of how a payment request can be initiated, it is a not a recommendation for how to implement the payment APIs in your own applications.

The Query Payment API Call

This call is used to discover the status of a previously executed purchase request. Typically, this call is made when a merchant or PSP queries the final outcome of a payment.

URLs:

Test

https://api.test.oneroadpayments.com/merchants/{*merchantID*}/transactions/{orderID}

Production

https://api.oneroadpayments.com/merchants/{*merchantID*}/transactions/{orderID}

Request Method:

GET

Headers:

The header '*x-api-key*' should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

Name	Description	M/O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
orderID	The order ID provided with the payment request which uniquely identifies a payment	M

Body- Query Payment Request

NA- No body data required.

The query payment request should be sent to the MYGW server using the GET method.

Body – Query Payment Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful query payment request is as follows.


```
{
"response":{
   "statusCode":200,
   "message":"OK"
   },
"reply":
   {
   "amount":"25.00",
   "balance":"25.00",
   "clientID":"TheClient",
   "currency":"USD",
   "date":"2024-07-16T06:20:53.281",
   "detail":"Successfully completed",
   "merchant":"The Merchant",
   "merchantID":"5adeaafb-1b6d-4bb2-ba11-1cce35e6b38e",
   "orderID":"110836419",
   "reason":"Success!",
   "result":"0",
   "settlement":"2024-07-23",
   "signature":"3a53e1e7251b08036cc2f9b8de9d2030",
   "success":"Y",
   "url":"https://www.my-
gateway.net/ReturnURL.html?currency=AUD&success=Y&merchantID=5adeaafb-
1b6d-4bb2-ba11-
1cce35e6b38e&orderID=110836419&clientID=NewClient&amount=25.00&signature
=3a53e1e7251b08036cc2f9b8de9d2030"
   }
}
```


The details of the response are in the following table, M means a value that is always returned with a successful query payment response; and O means a value which may be returned depending on the status of the original payment request.

Name	Description	M/O
clientID	Identifies the client who performed the original payment transaction	O
amount	The amount of the purchase transaction	M

balance	The amount still pending settlement to the merchant	O
currency	The transaction currency for the original payment transaction	M
date	The date on which the original payment transaction was completed	O
orderID	The unique merchant order number for the original transaction	M
URL	Details of the callback URL performed for this transaction	O
merchant	The name of the merchant as stored with the gateway	O
merchantID	The unique merchant ID provided by the PSP or allocated by the MYGW system	M
detail	More information on the status of the payment transaction	O
reason	More information on the status of the payment transaction	O
result	A numeric identifier which denotes the final result of the transaction, can be one of the values shown below	O
settlement	If the payment transaction completed successfully, this is the date on which the funds will be transferred to the merchant	O
signature	The cryptographic hash for the payment transaction results, created using the *Merchant Key*, calculated as described above under *Callback URL Processing*	M

The value of the result field can be one of the following, the meanings of which have been described above.

0 – Transaction completed successfully
1 – Transaction pending, incomplete
2 – Signature mismatch
4 – Client record not found
5 – Duplicate order identifier
7 – Unable to determine the gateway processor
9 – Transaction declined, the transaction was declined by the account or card issuer

10 – Amount out of range for this merchant
11 – Transaction abandoned, the transaction was not completed within the timeout period
15 – Currency not supported for this merchant
17 – Blocked due to risk score

An example of the JSON object sent in a response to an unsuccessful query payment request is as follows.


```
{
"response":
   {
   "statusCode": 404,
   "message": " "Not Found",
   "detail": " No OrderID found with MerchantID: 5adeaafb-1b6d-4bb2-ba11-1cce35e6b38e"
   }
}
```


The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The query payment result was successfully returned from the server	*Message* - OK
400	The request URL was incorrectly formatted	*Message* - Bad request *Detail* – Bad parameters
403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
404	No order with the ID specified was found for the merchant at the server	*Message* – Not Found *Detail* – No Order found with MerchantID
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Query Payment - Sample Source Code

An example of executable JavaScript sample source code to query a payment can be found by clicking on the icon below. Note that this is an example of how a query payment request can be initiated, it is a not a recommendation for how to implement the query API in your own applications.

Refunds and Cancels /Voids

Refunds

For any goods purchased with a card that are accepted for return, or for any services that are terminated or cancelled, or where any price adjustment is made, merchants may need to refund the transaction amount to the cardholder's account. Card schemes require a refund to only be processed on the same card number that was used in the original sales transaction and for an amount which is no more than the original transaction.

Cancel/Void

Merchants and PSPs are able to cancel or void a transaction which has been authorised by the card issuer prior to settlement. A transaction may need to be cancelled if the cardholder completes a payment transaction in error, for the wrong amount or would prefer to choose a different method of payment. Cancels can only be processed on previously authorised transactions which have not yet been cleared by the card issuers and a cancellation will always be performed for the entire transaction amount.

Manually Performed Cancel and Refund Transactions

Cancel and refund transactions can be performed using the *Merchant Administration System,* described in the relevant user interface document or via an API call from the merchant or processor site as described below.

Some merchants and PSPs on the My Gateway Switch are only able to perform refund and void transactions using the *Merchant Administration System*. Those merchants and PSPs are not able to use the refund and void API calls.

The Refund and Cancel API Calls

These calls are used to

- refund part or all of a previously completed transaction on the My Gateway Switch after the merchant has been settled
- cancel all of a purchase made prior to the to the merchant being settled

URLs:

Test
https://api.test.oneroadpayments.com/commands/
Production
https://api.oneroadpayments.com/commands/

Request Method:

POST

Headers:

The header '*x-api-key*' should be set to the value of the Merchant API Key allocated to the merchant or PSP at sign up.

URL Parameters:

None

Body – Refund/Cancel Request

Name	Description	R/M/O
command	Always set to 'BackOffice' for cancel and refund requests	M
p1	Set to 'Refund' to refund a transaction and 'Cancel' to cancel a transaction	M
p2	The merchant ID of the merchant. The unique merchant ID provided by the PSP or allocated by the MYGW system	M
p3	The unique order number for the original transaction to be refunded or cancelled	M
p4	The date to process the refund or cancel, in the format YYYY-MM-DD	M
p5	The amount to be cancelled or refunded. The refund amount can be less than or equal to the amount of the original transaction; the cancel amount must be the same as the original transaction amount.	M

The refund or cancel request data should be sent to the MYGW server formatted as a JSON object, using the POST method.

Refund/Cancel Response

The MYGW server will respond including a status code, signifying whether the call succeeded, as described above - *MYGW API Responses*.

An example of the JSON object sent in a response to a successful refund or cancel request is as follows.

```
{
"response":
    {
    "statusCode": 200,
    "message": "OK"
    },
}
```

An example of the JSON object sent in a response to an unsuccessful refund or cancel request is as follows.

```
{
"response":
   {
   "statusCode":400,
   "message":"Bad Request",
   "detail":"Bad Command"
   }
}
```

The status code and status message returned could be one of the following values.

Status Code	Description	Status Message
200	The refund/cancel was successfully executed or queued for processing	*Message* - OK
400	The request was not correctly formatted	*Message* - Bad Request *Detail* – Bad Command
400	Either the merchant ID or order ID were not valid	*Message* - Bad Request *Detail* – Refund/void failed Original transaction not found
400	The amount in the request is invalid	*Message* - Bad Request *Detail* – Refund/void failed Amount not valid

403	The merchant API key could not be validated, and the request failed	*Message* - Forbidden
500	A server error occurred, and the request failed	*Message* – Internal processing exception

Refund/Cancel - Sample Source Code

An example of executable JavaScript sample source code to refund or cancel a transaction can be found by clicking on the icon below. Note that this is an example of how a refund or cancel can be initiated, it is a not a recommendation for how to implement the API in your own applications.